



INTERNATIONAL CURRICULUM CENTRE

RENDA FUZHONG



IBDP Computer Science      Major Assessment 5  
Mock Final Exam      Mark Scheme

Name:

Marks:

Total Marks: 82 + 20(Extra marks)

Exam Time: 100 minutes

Please answer all the questions on the question paper. You are not allowed to use any electronic devices.

There are two papers in this exam. You should answer ALL the questions.

**Paper 1: Core, 8 MCQ + 2 Short answer question + 1 Structured question [37 marks in total]**

1 Which of the following statements about recursion are true?

- I Every recursive algorithm can be written iteratively.
- II Tail recursion is always used in “divide-and-conquer” algorithms.
- III In a recursive definition, a process is defined in terms of a simpler case of itself.

- (A) I only
- (B) III only
- (C) I and II only
- (D) I and III only
- (E) II and III only

D

1. (D) Tail recursion is when the recursive call of a method is made as the last executable step of the method. Divide-and-conquer algorithms like those used in mergesort or quicksort have recursive calls *before* the last step. Thus, statement II is false.

- 2 The method `changeNegs` below should replace every occurrence of a negative integer in its matrix parameter with 0.

```
/** @param mat the matrix
 *   Precondition: mat is initialized with integers.
 *   Postcondition: All negative values in mat replaced with 0.
 */
public static void changeNegs(int[] [] mat)
{
    /* code */
}
```

Which is correct replacement for `/* code */`?

```
I for (int r = 0; r < mat.length; r++)
    for (int c = 0; c < mat[r].length; c++)
        if (mat[r][c] < 0)
            mat[r][c] = 0;

II for (int c = 0; c < mat[0].length; c++)
    for (int r = 0; r < mat.length; r++)
        if (mat[r][c] < 0)
            mat[r][c] = 0;

III for (int[] row : mat)
    for (int element : row)
        if (element < 0)
            element = 0;
```

- (A) I only
- (B) II only
- (C) III only
- (D) I and II only
- (E) I, II, and III

D

(D) Segment I is a row-by-row traversal; segment II is a column-by-column traversal. Each achieves the correct postcondition. Segment III traverses the matrix but does not alter it. All that is changed is the local variable `element`. You cannot use this kind of loop to replace elements in an array.

- 3 A feature of data that is used for a binary search but not necessarily used for a sequential search is
- (A) length of list.
  - (B) type of data.
  - (C) order of data.
  - (D) smallest value in the list.
  - (E) median value of the data.

(C) The binary search algorithm depends on the array being sorted. Sequential search has no ordering requirement. Both depend on choice A, the length of the list, while the other choices are irrelevant to both algorithms.

- 4 Suppose that base-2 (binary) numbers and base-16 (hexadecimal) numbers can be denoted with subscripts, as shown below:

$$2A_{\text{hex}} = 101010_{\text{bin}}$$

Which is equal to  $3D_{\text{hex}}$ ?

- (A)  $111101_{\text{bin}}$
- (B)  $101111_{\text{bin}}$
- (C)  $10011_{\text{bin}}$
- (D)  $110100_{\text{bin}}$
- (E)  $101101_{\text{bin}}$

A

- 5 Assume that a square matrix `mat` is defined by

```
int[] [] mat = new int[SIZE][SIZE];
//SIZE is an integer constant >= 2
```

What does the following code segment do?

```
for (int i = 0; i < SIZE - 1; i++)
    for (int j = 0; j < SIZE - i - 1; j++)
        swap(mat, i, j, SIZE - j - 1, SIZE - i - 1);
```

You may assume the existence of this `swap` method:

```
/** Interchange mat[a][b] and mat[c][d]. */
public void swap(int[] [] mat, int a, int b, int c, int d)
```

- (A) Reflects `mat` through its major diagonal. For example,

$$\begin{array}{cc} 2 & 6 \\ 4 & 3 \end{array} \longrightarrow \begin{array}{cc} 2 & 4 \\ 6 & 3 \end{array}$$

- (B) Reflects `mat` through its minor diagonal. For example,

$$\begin{array}{cc} 2 & 6 \\ 4 & 3 \end{array} \longrightarrow \begin{array}{cc} 3 & 6 \\ 4 & 2 \end{array}$$

- (C) Reflects `mat` through a horizontal line of symmetry. For example,

$$\begin{array}{cc} 2 & 6 \\ 4 & 3 \end{array} \longrightarrow \begin{array}{cc} 4 & 3 \\ 2 & 6 \end{array}$$

- (D) Reflects `mat` through a vertical line of symmetry. For example,

$$\begin{array}{cc} 2 & 6 \\ 4 & 3 \end{array} \longrightarrow \begin{array}{cc} 6 & 2 \\ 3 & 4 \end{array}$$

- (E) Leaves `mat` unchanged.

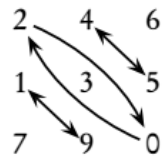
(B) Hand execute this for a  $2 \times 2$  matrix.  $i$  goes from 0 to 0,  $j$  goes from 0 to 0, so the only interchange is swap  $\text{mat}[0][0]$  with  $\text{mat}[1][1]$ , which suggests choice B. Check with a  $3 \times 3$  matrix:

```

i = 0  j = 0  swap mat[0][0] with mat[2][2]
          j = 1  swap mat[0][1] with mat[1][2]
i = 1  j = 0  swap mat[1][0] with mat[2][1]

```

The elements to be interchanged are shown paired in the following figure. The result will be a reflection through the minor diagonal.



- 6 A simple Tic-Tac-Toe board is a  $3 \times 3$  array filled with either X's, O's, or blanks. Here is a class for a game of Tic-Tac-Toe:

```
public class TicTacToe
{
    private String[] [] board;
    private static final int ROWS = 3;
    private static final int COLS = 3;

    /** Construct an empty board. */
    public TicTacToe()
    {
        board = new String[ROWS][COLS];
        for (int r = 0; r < ROWS; r++)
            for (int c = 0; c < COLS; c++)
                board[r][c] = " ";
    }

    /** @param r the row number
     *   @param c the column number
     *   @param symbol the symbol to be placed on board[r][c]
     *   * Precondition: The square board[r][c] is empty.
     *   * Postcondition: symbol placed in that square.
     */
    public void makeMove(int r, int c, String symbol)
    {
        board[r][c] = symbol;
    }

    /** Creates a string representation of the board, e.g.
     *   * |o |
     *   * |xx|
     *   * | o|
     *   * @return the string representation of board
     */
    public String toString()
    {
        String s = ""; //empty string
        /* more code */
        return s;
    }
}
```

X		
	O	
X		O

Which segment represents a correct replacement for `/* more code */` for the `toString` method?

- (A) 

```
for (int r = 0; r < ROWS; r++)
{
    for (int c = 0; c < COLS; c++)
    {
        s = s + "|";
        s = s + board[r][c];
        s = s + "|\\n";
    }
}
```
- (B) 

```
for (int r = 0; r < ROWS; r++)
{
    s = s + "|";
    for (int c = 0; c < COLS; c++)
    {
        s = s + board[r][c];
        s = s + "|\\n";
    }
}
```

```

(C) for (int r = 0; r < ROWS; r++)
{
    s = s + "|";
    for (int c = 0; c < COLS; c++)
        s = s + board[r][c];
}
s = s + "\n";

(D) for (int r = 0; r < ROWS; r++)
    s = s + "|";
for (int c = 0; c < COLS; c++)
{
    s = s + board[r][c];
    s = s + "\n";
}

(E) for (int r = 0; r < ROWS; r++)
{
    s = s + "|";
    for (int c = 0; c < COLS; c++)
        s = s + board[r][c];
    s = s + "\n";
}

```

(E) There are three things that must be done in each row:

- Add an opening boundary line:

```
s = s + "|";
```

- Add the symbol in each square:

```

for (int c = 0; c < COLS; c++)
    s = s + board[r][c];

```

- Add a closing boundary line and go to the next line:

```
s = s + "\n";
```

All of these statements must therefore be enclosed in the outer for loop, that is,

```
for (int r = ...)
```

A Deck class contains an array `cards` with an even number of Card values and a final variable `NUMCARDS`, which is an odd integer.

- 7      6. Here are two possible algorithms for shuffling the deck.

**Algorithm 1**

```
Initialize an array of Card called shuffled of length NUMCARDS.
Set k to 0.
For j=0 to NUMCARDS/2-1
    - Copy cards[j] to shuffled[k]
    - Set k to k+2
Set k to 1.
For j=NUMCARDS/2 to NUMCARDS-1
    - Copy cards[j] to shuffled[k]
    - Set k to k+2
```

**Algorithm 2**

```
Initialize an array of Card called shuffled containing NUMCARDS slots.
For k=0 to NUMCARDS-1
    - Repeatedly generate a random integer j from 0 to NUMCARDS-1,
      until cards[j] contains a card not marked as empty
    - Copy cards[j] to shuffled[k]
    - Set cards[j] to empty
```

Which is a *false* statement concerning Algorithms 1 and 2?

- (A) A disadvantage of Algorithm 1 is that it won't generate all possible deck permutations.
- (B) For Algorithm 2, to determine the last element shuffled requires an average of `NUMCARDS` calls to the random number generator.
- (C) Algorithm 2 will lead to more permutations of the deck than Algorithm 1.
- (D) In terms of run time, Algorithm 2 is more efficient than Algorithm 1.
- (E) If Algorithm 1 is repeated several times, it may return the deck to its original state.

**(D)** The big defect of Algorithm 2 is that it eventually slows down. This is because every time it selects an empty element, it has to loop again. Each of the other choices is true. In choice A, for example, the element `cards[0]` always moves to `shuffled[0]`, eliminating all permutations that have `cards[0]` in a different slot. For choice B, by the time you get to assign the last element, all but two slots of the `cards` array are marked empty. So, on average, you will need to go through `NUMCARDS` tries to find one of those two nonempty slots. For choice C, even though Algorithm 2 is slow, in theory every element in `cards` could land in any given slot in `shuffled`. This is not true for Algorithm 1, where the first element never budes out of the first slot. For choice E, because of the precise ordering of elements in Algorithm 1, the array will always eventually return to its original state, assuming there are sufficient iterations.

- 8 The following shuffle method is used to shuffle the cards in the Deck class.

```

Line 1: public void shuffle()
Line 2: {
Line 3:     for (int k = NUMCARDS; k > 0; k--)
Line 4:     {
Line 5:         int randPos = (int) (Math.random() * (k + 1));
Line 6:         //swap randomly selected card with card at position k
Line 7:         Card temp = cards[k];
Line 8:         cards[k] = cards[randPos];
Line 9:         cards[randPos] = temp;
Line 10:     }
Line 11: }

```

The method does not work as intended. Which of the following changes should be made to correct the method?

- (A) Replace Line 3 with  
for (int k = NUMCARDS; k >= 0; k--)
- (B) Replace Line 3 with  
for (int k = NUMCARDS - 1; k > 0; k--)
- (C) Replace Line 3 with  
for (int k = 1; k <= NUMCARDS; k++)
- (D) Replace Line 5 with  
int randPos = (int) (Math.random() \* k);
- (E) Replace Lines 7 – 9 with  
Card temp = cards[randPos];  
cards[randPos] = cards[k];  
cards[k] = temp;

(B) If k starts with the value NUMCARDS, the method encounters cards [NUMCARDS] on Line 7 and throws an `ArrayIndexOutOfBoundsException`.

MCQ answer table

1	2	3	4	5	6	7	8
D	D	C	A	B	E	D	B

### Short answer questions

- 9 Use a selection sort to put the following set of numbers into order from highest to lowest. List the results after each pass.

12 52 16 42 88 86

[3]

Award marks as follows up to **[3 marks max]**.

Award **[2 marks max]** for the first three passes correct,

(**[1 mark]** for at least one of passes 1, 2 and 3 correct).

Award **[1 mark]** for correct passes 4 and 5 with no change on pass 4.

Pass	12	52	16	42	88	86
1	88	52	16	42	12	86
2	88	86	16	42	12	52
3	88	86	52	42	12	16
4	88	86	52	42	12	16
5	88	86	52	42	16	12

[3]



10 Consider the following method.

```
public int GCD (int x, int y) // x,y not both 0
{
    if (y==0) return x;
    return GCD(y,x%y);
}
```

Trace the method to compute  $\text{GCD}(15, 20)$ , by copying and completing the table below. [3 marks]

x	y	return
15	20	...

*Award [1 mark] for correct column x.*

*Award [1 mark] for correct column y.*

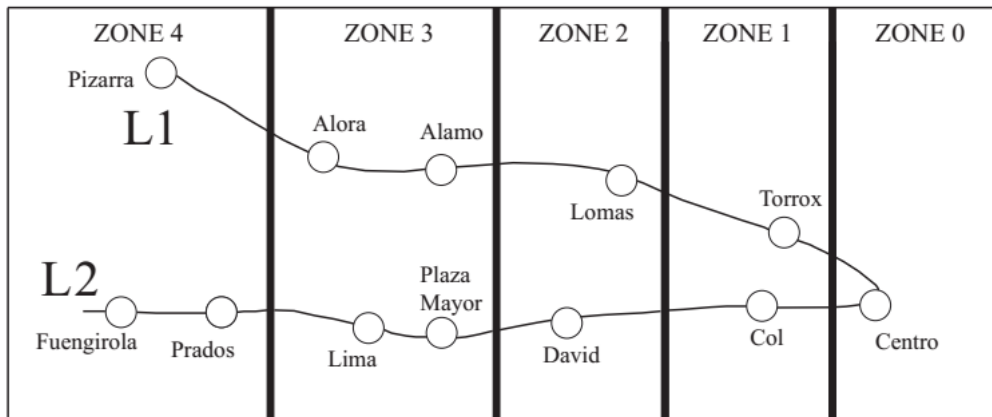
*Award [1 mark] for the correct final value (5) returned.*

x	y	return
15	20	GCD(20,15)
20	15	GCD(15,5)
15	5	GCD(5,0)
5	0	5

[3 marks]

### Structured questions:

- 11 A suburban railway system for a large city in Southern Europe consists of two lines **L1** and **L2**, which meet at the station Centro, where passengers can change from one line to the other. The system is shown below.



Each station is located in a particular zone, and the total number of zones in which the journey takes place determines the train fare. Note, if a passenger starts in **Zone 1**, goes to **Zone 0** and then back to **Zone 1**, the journey has taken place in **three** zones. Examples of the number of zones are shown below for different journeys.

Travelling from	Travelling to	Number of zones
Lima	Plaza Mayor	1
Alora	Plaza Mayor	7
Lomas	Col	4

- (a) State the number of zones in which the journey takes place when travelling from Alora to Fuengirola. [1]

The data for each station (station name, line, zone) is stored on the system's server in the collection `TRAIN_DATA`. There are 12 stations in total. The first part of the collection is shown below.

Centro, L1, 0, Alora, L1, 3, Torrox, L1, 1, Col, L2, 1, ...

From this we can see that Alora is part of line L1 and is located in Zone 3.

At the start of each day, the data in `TRAIN_DATA` is read in to the binary tree `TREE`, in which each node will hold the data for one station. The binary tree will be used to search for a specific station's name.

- (b) Sketch the binary tree after the station data from the first part of the collection, given above, has been added. [3]

The `TRAIN_DATA` collection is also used to construct the one-dimensional array `STATIONS` (which only contains the list of station names sorted into alphabetical order), where `STATIONS[0] = Alamo`.

- (c) State the value of `STATIONS[4]`. [1]

The two data structures (`STATIONS` and `TREE`) are now used to construct the two-dimensional array `FARES` containing the fares between stations, partly shown below. Note that the fare for travelling in **each** zone is €1.00.

<b>FARES</b>	Alamo	Alora	Centro	Col	...
Alamo	0	1.00	4.00	5.00	...
Alora	1.00	0	4.00	5.00	...
Centro	4.00	4.00	0	2.00	...
Col	5.00	5.00	2.00	0	...
...	...	...	...	...	...etc

- (d) Calculate the fare for travelling from Torrox to Lima. [1]

- (e) Construct the algorithm that would calculate the fares for this two-dimensional array. You can make use of the following two sub-procedures:

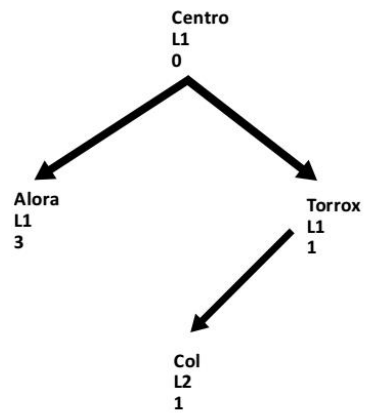
- `TREE.getZone(STATION)` // which returns the zone in which the  
// station is located
- `TREE.getLine(STATION)` // which returns the line on which the  
// station is located

Your algorithm should make as few calculations as possible. [9]

(a) 8;

**[1 mark]**

(b) *Centro* as root;  
Station names in correct position;  
All 3 items of data for each node;



**[3 marks]**

(c) David;

**[1 mark]**

(d) 5.00 (Euros);  
*Accept 5.*

**[1 mark]**

- (e) Award **[1 mark]** for each of the following 11 points, up to **[9 marks max]**.
- use of nested loops;
  - use of nested loops with indices that avoid repeating calculations (as shown);  
(Note: outer loop can be to 11 if repeat calculations are avoided, with an IF statement)
  - correct values retrieved from tree;
  - check for same line;
  - check if one of the stations is "Centro";
  - check and change if negative/ use of absolute value;
  - correct calculation for same line/one station is "Centro";
  - correct calculation for different line;
  - assign value to array;
  - assign mirror value;
  - assign value to diagonal;

```

loop N from 0 to 10
    STATION1 = STATION[N]
    AZ = TREE.getZone[STATION1]
    AL = TREE.getLine[STATION1]
    loop M from N+1 to 11 //start index changed so as not to repeat
        //code
        STATION2 = STATION[M]
        BZ = TREE.getZone[STATION2]
        BL = TREE.getLine[STATION2]
        if AL = BL or STATION1 = "Centro" or STATION2 = "Centro"
            then
                //on same line or passing through "Centro"
                X = AZ - BZ //number of zones where the travel takes
                //place can be negative
                if X<0 then //allow use of absolute
                    X = -X //or equivalent,e.g. X = abs(AZ-BZ)
                endif
                X = X+1
            else //on different lines
                X = AZ+BZ+1
            endif
            FARES[N][M]=X //assigns value to 2D array
            FARES[M][N]=X //assigns mirror value
        endloop
    FARES[N][N]=0 //leading diagonal
endloop
FARES[11][11]=0 //final entry

```

**[9 marks]**

**Total: [15 marks]**

**Paper 2: Option D: OOP, 3 structured questions.**

**[45 marks in total (Question 12,13,14) + 20 Extra marks ( Question 15) ]**

**Option D — Object-oriented programming**

A library in a college loans books to students.

The current book borrowing system, which was developed years ago, is prone to error. You are a member of the new program development team which has decided to develop a system using an object-oriented programming (OOP) approach. The team will be divided into small sub-groups.

The program development team is aware of the following user requirements.

- Certain books are classified as **short term** and can only be borrowed for 5 days. Other books are classified as **long term** and can be borrowed for 30 days.
- Fines are charged at a rate of \$1.50 for each day that a short term book is overdue.
- Fines are charged at a rate of 10 cents for each day that a long term book is overdue.
- If the overdue period exceeds 4 days for any short term loan book students are not allowed to borrow any more books and their borrowing rights are suspended; this does not apply for long term book loans.
- Students can borrow a total of 10 books.

15. (a) Outline **two** advantages that the programming team should expect from using an OOP approach. [4]

12 (b) Explain **two** ethical issues which should be taken into account when working on the project. [4]

(a) *Award up to [4 max].  
Award [1] for an advantage and [1] for an elaboration for up to two advantages.*

Re-use objects/modules across sub-groups;  
Which will speed up development;  
Work can be split up across the sub-groups;  
Which will speed up development;  
Sub-groups can re-use objects;  
Without having to re-program;  
Quicker to test;  
As each module is small/independent;

**[4]**

(b) *Award up to [4 max].  
Award [1] for an issue and [1] for an explanation stating its importance, for two issues, up to [4 max].*

Testing;  
Of system is extremely important to see that loans, fines, borrowing and rights work correctly;  
(Any one of the reasons on the list, not all required)  
Acknowledgement of code sources;  
Important to avoid any potential legal issues;  
*Accept other reasonable explanations that clearly refer to ethical aspects for the work of the programmers.*

**[4]**

Following some analysis, the team determined that the Loan and Student classes will be used. Part of the Student class is shown below.

```
import java.util.*;
public class Student
{
    private int studentID;
    private String studentName;
    private Loan[] booksBorrowed = new Loan[10];
    private int numBooks = 0;
    public Student(int studentID, String studentName)
    {
        this.studentID=studentID;
        this.studentName=studentName;
    }
    public Loan getLoan (int x)
    {
        return this.booksBorrowed[x];
    }
    public void addLoan(Loan book)
    {
        this.booksBorrowed[numBooks] = book;
        numBooks++;
    }
    public int getStudentID()
    {
        return this.studentID;
    }
    public String getStudentName()
    {
        return this.studentName;
    }
}
```

When creating objects, encapsulation is an important design consideration.

- (c) Outline, with direct reference to the `Student` class, how security can be enhanced by using encapsulation. [3]

To use the `Student` class in a program an object has to be created or instantiated.

- (d) Construct a statement to create a `Student` object for a student with an ID of 93003 and a name of "Smith". [2]

- |   |            |
|---|------------|
| <p>(c) <b>Award up to [3 max].</b><br/>                 By using private for variables for example studentID;<br/>                 Data can be protected from accidental changes;<br/>                 From outside the object;</p> | <b>[3]</b> |
| <p>(d) <b>Award up to [2 max].</b><br/>                 Student newStudent = new Student(93003, "Smith")<br/> <b>Award [1]</b> for Student newStudent<br/> <b>Award [1]</b> for new Student(93003, "Smith").</p>                    | <b>[2]</b> |

The basic `Loan` class is shown below. Every time a book is borrowed, an instance of the `Loan` class is created. In the `Student` class, the books borrowed by a single student are stored in an array called `booksBorrowed`.

```
import java.util.*;
public class Loan
{
    private int bookID;
    private String bookTitle;
    private Date d;
    static int numBooksLoaned = 0;
    public Loan(int bookID, String bookTitle)
    {
        this.bookID = bookID;
        this.bookTitle = bookTitle;
        this.d = new Date(); //set date borrowed
        numBooksLoaned = numBooksLoaned + 1;
    }
    public int getBookID()
    {
        return this.bookID;
    }
    public String getBookTitle()
    {
        return this.bookTitle;
    }
    public Date getDate()
    {
        return this.d;
    }
    public void setBookID(int id)
    {
        this.bookID = id;
    }
    public void setBookTitle(String title)
    {
        this.bookTitle = title;
    }
    public void setDate(Date d)
    {
        this.d = d;
    }
}
```

Note the use of the keyword `static` in the statement `static int numBooksLoaned = 0` and that this term is not used in the other variable declarations.

- (e) Explain the use of the term `static` for the variable `numBooksLoaned` in the `Loan` class. [3]
- (f) Construct the code needed to add a loan with a book ID of 212000 and a book title of "The Stars" to a `Student` object, `ST`. [3]

- (e) **Award up to [3 max].**  
 Static means that the variable is declared once only;  
 Static variable is a class variable not re-declared in each object;  
 And not each time an object is created;  
 The variable represents the number of times a `Loan` object is created;  
 And hence is a total of how many books have been borrowed; [3]
- (f) **Award up to [3 max].**  
`ST.addLoan (new Loan(212000, "The Stars"))`  
**Award [1] for `ST.addLoan`.**  
**Award [1] for `new`.**  
**Award [1] for `Loan (212000, "The Stars")`.** [3]



- 13 In order to store each `Student` object the team has decided to use the `borrowers` array of type `Student`. The team has also decided to use the student's ID as the index into the array. For example, a `Student` object with a student ID of 1111 would have its reference stored in position 1111 of the array.

Consider the code fragment below which would appear in a suitably constructed `main` class.

```
public static void main(String[] args)
{
    Student temp;
    Student[] borrowers = new Student[100000];

    temp = new Student(93001, "Jones");
    temp.addLoan(new Loan(210001, "The Sky"));
    borrowers[93001] = temp;

    temp = new Student(3012, "Zang");
    temp.addLoan(new Loan(210121, "The Animals"));
    borrowers[3012] = temp;

    borrowers[93001].addLoan(new Loan(210002, "The Spooks"));

    temp = new Student(93002, "Nguyen");
    temp.addLoan(new Loan(210011, "The Ocean"));
    borrowers[93002] = temp;

    System.out.println(borrowers[93001].getStudentName());
    System.out.println(borrowers[93001].getLoan(1).getBookTitle());
    System.out.println(borrowers[3012].getLoan(0).getBookTitle());
}
```

- (a) Determine the output from the following statements.

- |   |     |
|---|-----|
| (i) <code>System.out.println(borrowers[93001].getStudentName());</code>           | [1] |
| (ii) <code>System.out.println(borrowers[93001].getLoan(1).getBookTitle());</code> | [1] |
| (iii) <code>System.out.println(borrowers[3012].getLoan(0).getBookTitle());</code> | [1] |

The librarian requires a display of the student name, the titles of the books and the dates that they were borrowed, corresponding to a specific student ID.

- |  |     |
|--|-----|
| (b)    Construct the method <code>showDetails()</code> that could be used in the <code>main</code> class, which accepts a student ID as a parameter, and then uses this to access the appropriate <code>Student</code> object producing the desired output as indicated above. | [7] |
| (c)    State <b>three</b> aspects of the user requirements that are not addressed by the current design.   | [3] |
| (d)    Describe a modification to the <code>Student</code> class that would address <b>one</b> of the missing requirements.  | [2] |

- (a) (i) Jones; [1]  
 (ii) The Spooks; [1]  
 (iii) The Animals; [1]

- (b) **Award up to [7 max].**  
*Award [1] for correct signature (do not accept non-void/return methods. StudentID must be passed as a parameter).*  
*Award [1] for correct identification of student s.*  
*Award [1] for printing student's name.*  
*Award [1] for correct loop with limit of  $i < 10$  or equivalent expressions.*  
*Award [1] for correct identification of current loan.*  
*Award [1] for test on loan.*  
*Award [1] for printout of both Title and Date of the current loaned book.*

*For example:*

```
public void showDetails (int StudentID )
{
    Student s = borrowers[StudentID];
    System.out.println(s. getStudentName());
    for (int i=0; i<10; i++)
    {
        Loan currentLoan = s.getLoan(i);
        if (currentLoan != null)
        {
            System.out.println(currentLoan.getBookTitle());
            System.out.println(currentLoan.getDate());
        }
    }
}
```

[7]

**Note:** Award [1] for **both final** System.out.println statements. Marks are not awarded if methods are used that are not defined or already present in the specification.

- (c) **Award up to [3 max].**  
 There is no way to determine the fines;  
 There is no way to differentiate between the short term and long term loan books;  
 There is no way to indicate if a student's borrowing rights are suspended;  
 Nothing to prevent error or check number of books borrowed [3]

- (d) **Award up to [2 max] for any acceptable description.**

*For example:*

Add a Boolean variable to represent the borrowing rights;  
 Initially set this variable to true;  
 The variable is set to true unless rights are suspended when it is set to false;  
 etc; [2]

14

The team has decided to use inheritance in its design.

- (a) Describe **one** benefit that inheritance brings to OOP.

[2]

Two sub-classes will be created: `shortTerm` and `longTerm`.

- (b) State **two** additional attributes that each `shortTerm` object should have.

[2]

- (c) Construct a suitable UML diagram showing the relationships between the `Student` class, the `Loan` superclass and the two sub-classes. **Note:** there is no need to include the attributes or methods of each class.

[4]

- (d) Outline, using an OOP technique, how the total number of books on loan could be displayed without processing the `borrowers` array.

[3]

(a) Award up to **[2 max]**.  
Inheritance reduces the amount of coding / reduces repetition of code;  
By allowing sub-classes to inherit the methods / attributes of their superclass; [2]

(b) Award up to **[2 max]**.  
Variable for the fine (per day);  
Variable for the borrowing limit;  
Variable `timeLimit`; [2]

(c) A suitable diagram will look like this.

```

classDiagram
    class StudentClass
    class LoanClass
    class ShortClass
    class LongClass
    StudentClass "1" *-- "*" LoanClass
    LoanClass <|-- ShortClass
    LoanClass <|-- LongClass
  
```

Award **[1]** for all four correct classes as shown.  
Award **[1]** for showing the links in the correct place (any arrow or line).  
Award **[2]** for distinguishing either with labels (Student has a Loan / Short is a Loan) or use of different arrows for the different relationships between Student–Loan and Loan–Short/Long. [4]

(d) Award up to **[3 max]**.  
Add a static count variable;  
To the `shortTerm` class and to the `longTerm` class;  
A method can be called to display count by adding the two above together; [3]

- 15 The librarian has asked that the system be able to locate books quickly in order to find out who has borrowed a specific book.
- (a) Explain why it would be a slow process to locate borrowed books in the current system design. [4]
  - (b) (i) Describe a suitable data structure, which is referred to in the course (other than an array), that would speed up access to the ID of a student who has borrowed a specific book. [2]
  - (ii) Outline the steps involved in using this data structure that would allow the student ID of the person borrowing a specific book to be found quickly. [4]
- It has been decided to replace the `borrowers` array with a linked list data structure.
- (c) Describe **one** advantage and **one** disadvantage of replacing the original `borrowers` array with this structure. [4]
  - (d) Construct the algorithm, `addToEnd()`, that would allow a short term book to be added to the end of this data structure. [6]

- (a) *Award up to [4 max].*  
The books are stored in separate arrays;  
In separate objects;  
These can only be searched using a linear search;  
As no index is available;  
Also, parts of each array will be empty;  
So this would be slow/inefficient; [4]
- (b) (i) *Award up to [2 max].*  
A binary tree;  
Using the book ID as the index; [2]
- (ii) *Award up to [4 max].*  
Every time a loan object is created;  
Book object inserted using bookID;  
The student ID stored also;  
To locate the student ID the binary tree would be searched;  
Using the bookID;  
The studentID returned;  
Used to access the borrowers array to find the student name; [4]
- (c) *Award up to [4 max].*  
*Award [1] for an advantage and [1] for an elaboration, [1] for a disadvantage and [1] for an elaboration.*  
  
**Advantage:**  
There will be less wasted memory;  
As only the required number of locations will be used;  
  
**Disadvantage:**  
It will be slower to locate a specific student;  
As a linear search will be needed compared to direct access with the array; [4]
- (d) *Award up to [6 max]. Award [1] for each correct line of code (shown below; accept equivalents).*  
  
LinkedList p = new LinkedList ()  
LinkedList head = p.getHead()  
Set temp to head  
Moving through the list  
Recognizing that the last node has been reached  
Setting the shortTerm object to be the last and its pointer to null [6]